UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/388,766	09/02/1999	BOHR-WINN SHIH	303.513US1	4410

21186 7590 02/24/2005

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. BOX 2938
MINNEAPOLIS, MN 55402

EXAMINER	
SHARON, AYAL I	
ART UNIT	**PAPER NUMBER**
2123	

DATE MAILED: 02/24/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/388,766	SHIH ET AL.	
	Examiner	Art Unit	
	Ayal I Sharon	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 November 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *02 September 1999* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Introduction

1. Claims 1-35 of U.S. Application 09/388,766, originally filed on 09/02/1999 are presented for examination. The RCE filed 11/24/2004 contains amendments to claims 1, 8, 15, 20, 24, 31, 32, 33, and 34. No new claims were added, and no claims were cancelled. In addition, new arguments were presented.
2. Examiner notes that claims 28 and 29, while labeled as "previously presented", are not the claims presented in the previous amendment, dated 5/13/2004. The version of these claims presented in the current amendment is the version presented in the amendment filed on 8/20/2003.

Claim Interpretations

3. Examiner interprets a "node" as being an input or output to one or more gates in a circuit. Thus, Examiner interprets that the terms "signal" and "node" are interchangeable.
4. Examiner interprets a "forced" logic value as being one of the "forcing" values defined in the definition of the "std logic" package in IEEE Standard 1164-1993, p.2, which is held constant.
5. Applicants define the "release" of a node as meaning that "the simulation program is free to change the logic value of the node if [the] simulation warrants

a change in the logic value. Before the release, the logic value of the node is forced." (Specification, p.5)

6. Examiner interprets the "release of a node" as being equivalent to enabling a change in the logic value of the node.
7. Applicant defines "resolving" a node as meaning that "the inputs to the node are known and therefore the logic value of the node can be determined through simulation by the simulation program." (Specification, pp.5-6)
8. Examiner interprets "resolving" a node as having its value determined according to the resolution table and truth tables defined in IEEE Standard 1164-1993, pp.4-5.
9. Examiner interprets that one possible "predetermined condition" for "releasing" a node is a rising or falling edge of a clock pulse.
10. Examiner interprets "predetermined amount of time" as being the length of an entire clock cycle or a number of clock cycles.
11. Examiner interprets that a waveform plot (signal graph) is a form of "error indication" and "providing an indication when the node is in an undesirable condition", and "outputting a node condition".
12. Examiner interprets that a user-defined length of a clock cycle, or a user-defined number of clock cycles, constitutes a "user-defined time period".
13. Examiner interprets "therefrom" as being equivalent to "from this" or "from that".
14. Examiner interprets "conveyance" as being the input of a signal value.

Claim Rejections - 35 USC § 102

15. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office Action:

> A person shall be entitled to a patent unless-
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

16. The prior art cited is as follows:

17. IEEE Std 1364-1995. IEEE Standard Hardware Description Language Based on the Verilog® Hardware Description Language. Copyright 1995. (Henceforth referred to as "**IEEE 1364**". Sections 1,3,6,7,9,12,14,18,19,22, pp. i-ix, and the Index of this 675 page reference have included in the file wrapper and have been provided to the Applicants. The entire reference will be provided to the Applicants upon request.).

18. The claims are subsequently recited for Applicant's convenience. Applicant's attention is also directed to the pertinent sections of the prior art.

19. Claims 1-8, 12-30, and 33 are rejected under 35 U.S.C. 102(b) as being anticipated by IEEE 1364.

20. IEEE 1364 teaches the limitations of Claim 1:

1. (Currently Amended) A method of simulating a node using a simulation program that includes multiple, linked modules, the method comprising:

executing a first circuit module that simulates a circuit having a node, wherein the node represents a simulated electrical connection point of the circuit;
(See IEEE 1364, especially: Section 12, pp.135-139; Section 3, p.13; Section 6, pp.50-52;)

Example 1 (See Section 12, p.138) teaches an instance of a flip-flop module which defines variables that represent "output nets" and "input nets". Examiner interprets that these variables correspond to the claimed "electrical connection points of the circuit."

Section 3, p.13, teaches that "There are two main groups of data types: the register data types and the net data types."

Section 6, pp.50-52 describes, in greater detail, the mechanism for placing values into nets and registers.

simultaneously executing at least one behavior module, which is linked to the first circuit module, and which performs the acts of
(See IEEE 1364, especially: Section 12, pp.135-139; Section 9, pp.104-106)

Section 12 teaches the use of modules, and Section 9 teaches the use of "force and release procedural statements" as claimed below.

forcing an initial forced logic state on the node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the "Force Procedural Statement" taught in Section 9, pp.104-106, corresponds to the "Force" commands claimed in this claim.

after forcing, releasing the node from the initial forced logic state if a predetermined condition is met, which enables the simulation program to change a logic state of the node
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

monitoring the released node after the node has been released; and
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

providing an indication, in response to the monitoring, when the node is in a pre-selected condition.

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

21. IEEE 1364 teaches the limitations of Claim 2:

2. (Previously Presented) The method of claim 1, wherein forcing the initial forced logic state includes forcing to a logic zero, logic one or high-impedance.

(See IEEE 1364, especially: Section 3, p.13)

22. IEEE 1364 teaches the limitations of Claim 3:

3. (Original) The method of claim 1, wherein releasing the node further comprises determining that the condition is met after passage of a predetermined amount of time.

(See IEEE 1364, especially: Section 9, pp.106-107 and pp.114-119; Section 14, pp.187-196; Section 18, pp.234, 246-247;)

23. IEEE 1364 teaches the limitations of Claim 4:

4. (Previously Presented) The method of claim 3, wherein releasing the node further comprises determining that the condition is met when the node has been resolved.

(See IEEE 1364, especially: Section 3, p.17-19)

24. IEEE 1364 teaches the limitations of Claim 5:

5. (Previously Presented) The method of claim 1, wherein providing an indication includes indicating when the released node is in an unknown logic state.

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247; Section 3, p.13)

25. IEEE 1364 teaches the limitations of Claim 6:

6. (Previously Presented) The method of claim 1, further comprising providing an error indication when the release node is in a pre-selected condition.

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

26. IEEE 1364 teaches the limitations of Claim 7:

7. (Original) The method of claim 3, further comprising selecting a user-defined time period for the predetermined amount of time.

(See IEEE 1364, especially: Section 9, pp.106-107 and pp.114-119; Section 14, pp.187-196; Section 18, pp.234, 246-247;)

27. IEEE 1364 teaches the limitations of Claim 8:

8. (Currently Amended) A method of initializing and monitoring a simulated circuit node using a simulation program that includes multiple linked modules, the method comprising:

executing a first circuit module that simulates a circuit having a node, wherein the node represents a simulated electrical connection point of the circuit;
(See IEEE 1364, especially: Section 12, pp.135-139; Section 3, p.13; Section 6, pp.50-52;)

Example 1 (See Section 12, p.138) teaches an instance of a flip-flop module which defines variables that represent “output nets” and “input nets”. Examiner interprets that these variables correspond to the claimed “electrical connection points of the circuit.”

Section 3, p.13, teaches that “There are two main groups of data types: the register data types and the net data types.”

Section 6, pp.50-52 describes, in greater detail, the mechanism for placing values into nets and registers.

simultaneously executing at least one behavior module, which is linked to the first circuit module, and which performs the acts of
(See IEEE 1364, especially: Section 12, pp.135-139; Section 9, pp.104-106)

Section 12 teaches the use of modules, and Section 9 teaches the use of “force and release procedural statements” as claimed below.

obtaining an initial node condition for the node, wherein the initial node condition is a logic state;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the “Force Procedural Statement” taught in Section 9, pp.104-106, corresponds to the “Force” commands claimed in this claim.

forcing the node to the initial node condition;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the “Force Procedural Statement” taught in Section 9, pp.104-106, corresponds to the “Force” commands claimed in this claim.

after forcing, releasing the node from the initial node condition;
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

> testing the node for a valid condition <u>after the node has been released</u>;

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

> monitoring the node <u>after the node has been released</u>; and

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

> providing an indication, <u>in response to the monitoring,</u> when the node is in an undesirable condition.

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

28. IEEE 1364 teaches the limitations of Claim 12:

12. (Original) The method of claim 8, further comprising outputting the condition of the simulated node.

(See IEEE 1364, especially: Section 9, p.110; Section 14, pp.179-183; Section 18, pp.234, 246-247;)

29. IEEE 1364 teaches the limitations of Claim 13:

13. (Original) The method of claim 8, further comprising obtaining a simulation run time.

(See IEEE 1364, especially: Section 9, pp.106-107 and pp.114-119; Section 14, pp.183-196, 202-204; Section 18, pp.234, 246-247;)

30. IEEE 1364 teaches the limitations of Claim 14:

14. (Original) The method of claim 13, further comprising outputting a final node condition when the simulation run time is completed.
(See IEEE 1364, especially: Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;)

31. IEEE 1364 teaches the limitations of Claim 15:

15. (Currently Amended) A computer-readable medium having computer-executable instructions comprising:

at least one selectable circuit module, which when executed simulates a circuit having a node, wherein the node represents a simulated electrical connection point of the circuit; and
(See IEEE 1364, especially: Section 12, pp.135-139; Section 3, p.13; Section 6, pp.50-52;)

Example 1 (See Section 12, p.138) teaches an instance of a flip-flop module which defines variables that represent "output nets" and "input nets". Examiner interprets that these variables correspond to the claimed "electrical connection points of the circuit."

Section 3, p.13, teaches that "There are two main groups of data types: the register data types and the net data types."

Section 6, pp.50-52 describes, in greater detail, the mechanism for placing values into nets and registers.

at least one selectable behavior module, which is linkable to a circuit module, and which when executed results in <u>performing the following acts:</u>
(See IEEE 1364, especially: Section 12, pp.135-139; Section 9, pp.104-106)

Section 12 teaches the use of modules, and Section 9 teaches the use of "force and release procedural statements" as claimed below.

forcing an initial forced logic state on the node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the "Force Procedural Statement" taught in Section 9, pp.104-106, corresponds to the "Force" commands claimed in this claim.

<u>after forcing,</u> releasing the node from the <u>initial</u> forced logic state if a predetermined condition is met, which enables a simulation program to change a logic state of the node and
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

monitoring the node after the node has been released; and

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

providing an indication, in response to the monitoring, when the node is in a preselected condition.

(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

32. IEEE 1364 teaches the limitations of Claim 16:

16. (Previously Presented) The medium of claim 15, having further computer-executable instructions for forcing the initial forced logic state to a logic zero, logic one or high-impedance.

(See IEEE 1364, especially: Section 3, p.13; Section 9, pp.104-106)

33. IEEE 1364 teaches the limitations of Claim 17:

17. (Original) The medium of claim 15, having further computer-executable instructions for determining that the condition is met after passage of a predetermined amount of time.

(See IEEE 1364, especially: Section 9, pp.106-107 and pp.114-119; Section 14, pp.179-190, 183-196, 202-204; Section 18, pp.234, 246-247;)

34. IEEE 1364 teaches the limitations of Claim 18:

18. (Previously Presented) The medium of claim 15, having further computer-executable instructions for determining that the condition is met when the node a valid logic value.

(See IEEE 1364, especially: Section 3, p.13; Section 9, pp.106-113)

35. IEEE 1364 teaches the limitations of Claim 19:

19. (Previously Presented) The medium of claim 18, having further computer-executable instructions for indicating when the release node is in an unknown logic state.
(See IEEE 1364, especially: Section 3, p.13; Section 9, pp.104-106; Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

36. IEEE 1364 teaches the limitations of Claim 20:

20. (Currently Amended) A simulation module of a simulation program, the simulation module comprising:

an input means for inputting an initial node condition into a simulated circuit node of a circuit module linked with the simulation module, wherein the simulated circuit node represents a simulated electrical connection point of the circuit module;
(See IEEE 1364, especially: Section 12, pp.135-139; Section 3, p.13; Section 6, pp.50-52; Section 9, pp.104-106)

Section 12 teaches the use of modules, and Section 9 teaches the use of "force and release procedural statements" as claimed below.

Example 1 (See Section 12, p.138) teaches an instance of a flip-flop module which defines variables that represent "output nets" and "input nets". Examiner interprets that these variables correspond to the claimed "electrical connection points of the circuit."

Section 3, p.13, teaches that "There are two main groups of data types: the register data types and the net data types."

Section 6, pp.50-52 describes, in greater detail, the mechanism for placing values into nets and registers.

a conveying means for conveying the initial node condition to the simulated circuit node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the "Force Procedural Statement" taught in Section 9, pp.104-106, corresponds to the "Force" commands claimed in this claim.

release means for releasing the simulated circuit node from the initial node condition upon satisfaction of a condition, wherein releasing the simulated circuit node <u>occurs after conveying and</u> enables the simulation program to change a logic state of the simulated circuit node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force

statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

a monitoring means for monitoring the simulated circuit node for a node condition <u>after releasing the simulated circuit node from the initial node condition</u>; and
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

an output means, responsive to the monitoring means, for outputting an indication when the node condition is in an undesirable state.
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

37. IEEE 1364 teaches the limitations of Claim 21:

21. (Original) The module of claim 20, further comprising an output means for outputting the node condition.
(See IEEE 1364, especially: Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;)

38. IEEE 1364 teaches the limitations of Claim 22:

22. (Original) The module of claim 20, further comprising an input means for inputting a simulation run time.
(See IEEE 1364, especially: Sections 9.2, 9.3, pp.103, and 106-107 and pp.114-119; Section 14, pp.179-190, 183-196, 202-204; Section 18, pp.234, 246-247;)

39. IEEE 1364 teaches the limitations of Claim 23:

23. (Original) The module of claim 22, further comprising an output means for outputting a final node condition at completion of the simulation run time.
(See IEEE 1364, especially: Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;)

40. IEEE 1364 teaches the limitations of Claim 24:

24. (Currently Amended) A computerized system for initializing and monitoring a simulated circuit node, the system comprising:

a circuit simulation tool;
(See IEEE 1364, especially: Section 1.2)

at least one selectable circuit module, which when executed simulates a circuit having the simulated circuit node, wherein the simulated circuit node represents a simulated electrical connection point of the circuit; and
(See IEEE 1364, especially: Section 12, pp.135-139)

Section 12 teaches the use of modules, and Section 9 teaches the use of "force and release procedural statements" as claimed below.

Example 1 (See Section 12, p.138) teaches an instance of a flip-flop module which defines variables that represent "output nets" and "input nets". Examiner interprets that these variables correspond to the claimed "electrical connection points of the circuit."

Section 3, p.13, teaches that "There are two main groups of data types: the register data types and the net data types."

at least one selectable behavior module, which is linkable to a circuit module, and which includes
(See IEEE 1364, especially: Section 9, pp.98-99)

a first input means for inputting an initial node condition;
(See IEEE 1364, especially: Section 6, pp.50-52; Section 9, pp.104-106)

Section 6, pp.50-52 describes, in greater detail, the mechanism for placing values into nets and registers.

a conveying means for conveying the initial node condition to the simulated circuit node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the "Force Procedural Statement" taught in Section 9, pp.104-106, corresponds to the "Force" commands claimed in this claim.

a release means for releasing the initial node condition, wherein releasing the initial node condition <u>occurs after conveying and </u>enables the circuit simulation tool to change a logic state of the simulated circuit node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force

statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

a monitoring means for monitoring the simulated circuit node for a node condition after releasing the initial node condition;
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

a first output means for outputting an indication when the node condition is in an undesirable state, in response to the monitoring;
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

a second input means for inputting a simulation run time; and
(See IEEE 1364, especially: Sections 9.2, 9.3, pp.103, and 106-107 and pp.114-119; Section 14, pp.179-190, 183-196, 202-204; Section 18, pp.234, 246-247;)

a second output means for outputting a final node condition at completion of the simulation run time.
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

41. IEEE 1364 teaches the limitations of Claim 25:

25. (Currently Amended) An HDL initial condition module comprising:

a means for forcing a logic level on a simulated circuit node;
(See IEEE 1364, especially: Section 9, pp.104-106)

Examiner finds that the "Force Procedural Statement" taught in Section 9, pp.104-106, corresponds to the "Force" commands claimed in this claim.

a means for maintaining the logic level of the simulated circuit node until a release condition is met, wherein
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

the simulated circuit node represents a simulated electrical connection point of a simulated circuit and the simulated circuit is produced by an HDL circuit module that is linkable to the HDL initial condition module, and
(See IEEE 1364, especially: Section 12, pp.135-139)

Section 12 teaches the use of modules, and Section 9 teaches the use of "force and release procedural statements" as claimed below.

Example 1 (See Section 12, p.138) teaches an instance of a flip-flop module which defines variables that represent "output nets" and "input nets". Examiner interprets that these variables correspond to the claimed "electrical connection points of the circuit."

Section 3, p.13, teaches that "There are two main groups of data types: the register data types and the net data types."

a simulation program is able to change a logic state of the simulated circuit node after the release condition is met.
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

<u>a means for monitoring the simulated circuit node after the simulated circuit node has been released; and</u>
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

a means for providing an indication, in response to the monitoring, when the simulated circuit node is in a preselected condition.
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

42. IEEE 1364 teaches the limitations of Claim 26:

26. (Previously Presented) The module of claim 25 wherein the release condition is when a known logic state can be determined for the simulated circuit node.
(See IEEE 1364, especially: Section 9, pp.104-106, 110-111;
Section 14, pp.179-183; Section 18, pp.234, 246-247;)

43. IEEE 1364 teaches the limitations of Claim 27:

27. (Original) The module of claim 25 wherein the logic level is a value defined by an HDL executable simulation program.
(See IEEE 1364, especially: Section 9, pp.104-106)

44. IEEE 1364 teaches the limitations of Claim 28:

28. (Currently Amended) An HDL initial condition module comprising:

an initial condition release means, which enables a simulation program to change a logic state of a simulated circuit node after a release condition is met, wherein the simulated circuit node represents a simulated electrical connection point of a simulated circuit, and the simulated circuit is produced by an HDL circuit module that is linkable to the HDL initial condition module; and
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

a simulated circuit node error detection means, which monitors the simulated circuit node for a node condition.
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

45. IEEE 1364 teaches the limitations of Claim 29:

29. (Currently Amended) An HDL initial condition module comprising:

means for maintaining a logic level of a simulated circuit node for a predetermined period of time, wherein the simulated circuit node represents a simulated electrical connection point of a simulated circuit, and the simulated circuit is produced by an HDL circuit module that is linkable to the HDL initial condition module; and
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment takes place, except in the case where a procedural continuous assignment is active on the register."

means for releasing an initial condition after a release condition is met, wherein releasing the initial condition enables a simulation program to change the logic level of the simulated circuit node, and wherein the predetermined period of time is a simulation run time defined by an HDL simulation executable program.
(See IEEE 1364, especially: Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;)

Section 14, pp.179-180 teach that the monitoring commands provide indications regarding node status.

46. IEEE 1364 teaches the limitations of Claim 30:

30. (Original) The module of claim 29, wherein the predetermined period of time is a user-defined period of time.
(See IEEE 1364, especially: Section 9, pp.104-106)

Section 9, pp.104 teaches that "A force statement to a register shall override a procedural assignment or procedural continuous assignment that takes place on the register until a release procedural statement is executed on the register. After the release procedural statement is executed, the register shall not immediately change value (as would a net that is forced). The value specified in the force statement shall be maintained in the register until the next procedural assignment

takes place, except in the case where a procedural continuous assignment is active on the register."

47. In regards to Claim 33, IEEE 1364 teaches a simulated HDL circuit device (Section 1, pp. iii-iv, 1-4), a plurality of HDL modules (Section 12, pp.135-139), nodes (Section 6, pp.50-52), behavior modules (Section 9), assigning/de-assigning and forcing/releasing node values, and condition statements (Section 9, pp.104-106), monitoring node values (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), and outputting node values (Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;), and outputting an undesirable state (Section 9, p.111).

Claim Rejections - 35 USC § 103

48. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

49. The prior art cited is as follows:

50. IEEE Std 1364-1995. IEEE Standard Hardware Description Language Based on the Verilog® Hardware Description Language. Copyright 1995. (Henceforth referred to as "**IEEE 1364**". Sections 1,3,6,7,9,12,14,18,19,22, pp. i-ix, and the Index of this 675 page reference have included in the file wrapper and have been

provided to the Applicants. The entire reference will be provided to the Applicants upon request.).

51. IEEE 1164-1993. IEEE Standard 1164-1993: Multivalue Logic System for VHDL Model Interoperability. Copyright 1995. (Henceforth referred to as "**IEEE 1164**").

52. Smith, D. J. "VHDL & Verilog Compared & Contrasted – Plus Modeled Example Written in VHDL, Verilog and C." Proc. 33rd Design Automation Conf. June 3-7, 1996. pp.771-776. (Henceforth referred to as "**Smith**").

53. Berman, V. "Standard Verilog-VHDL Interoperability." Proc. VHDL Int'l Users Forum. May 1-4, 1994. pp.142-149. (Henceforth referred to as "**Berman**")

54. QuickLogic QuickNote #61 "Initializing Flip-flops in QuickWorks Simulations". Last Updated: 3/19/98. (Henceforth referred to as "**QuickNote #61**").

55. The claims are subsequently recited for Applicant's convenience. Applicant's attention is also directed to the pertinent sections of the prior art.

56. **Claim 1, 8, 15, 20, 25, and 28-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over QuickNote #61 in view of Official Notice.**

57. QuickNote #61 teaches the limitations of Claim 1:

1. (Currently Amended) A method of simulating a node using a simulation program that includes multiple, linked modules, the method comprising:

executing a first circuit module that simulates a circuit having a node, wherein the node represents a simulated electrical connection point of the circuit;
(See QuickNote #61, especially: "Reasons for Initializing Flip Flops")

simultaneously executing at least one behavior module, which is linked to the first circuit module, and which performs the acts of
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

The reference teaches that "If you want to set an internal node called DRAM_CONTROLX to a value of '1', for example, you can create a small Verilog block in the Verilog test-fixture (.TF) file used to simulate the design. Here is an example ..." The example shows that the behavior module is separate from the module representing the electrical connections in the flip-flop.

forcing an initial forced logic state on the node;

(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

<u>after forcing,</u> releasing the node from the <u>initial</u> forced logic state if a predetermined condition is met, which enables the simulation program to change a logic state of the node

(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

monitoring the released node after the node has been released; and

providing an indication, in response to the monitoring, when the node is in a pre-selected condition.

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may never leave the unknown state unless it is manually initialized." There is always a possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

58. QuickNote #61 teaches the limitations of Claim 8:

8. (Currently Amended) A method of initializing and monitoring a simulated circuit node using a simulation program that includes multiple linked modules, the method comprising:

executing a first circuit module that simulates a circuit having a node, wherein the node represents a simulated electrical connection point of the circuit;

(See QuickNote #61, especially: "Reasons for Initializing Flip Flops")

simultaneously executing at least one behavior module, which is linked to the first circuit module, and which performs the acts of
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

The reference teaches that "If you want to set an internal node called DRAM_CONTROLX to a value of '1', for example, you can create a small Verilog block in the Verilog test-fixture (.TF) file used to simulate the design. Here is an example ..." The example shows that the behavior module is separate from the module representing the electrical connections in the flip-flop.

obtaining an initial node condition for the node, wherein the initial node condition is a logic state;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

forcing the node to the initial node condition;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

after forcing, releasing the node from the initial node condition;
(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

testing the node for a valid condition after the node has been released;

monitoring the node after the node has been released; and

providing an indication, in response to the monitoring, when the node is in an undesirable condition.

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may never leave the unknown state unless it is manually initialized." There is always a possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

59. QuickNote #61 teaches the limitations of Claim 15:

15. (Currently Amended) A computer-readable medium having computer-executable instructions comprising:

at least one selectable circuit module, which when executed simulates a circuit having a node, wherein the node represents a simulated electrical connection point of the circuit; and
(See QuickNote #61, especially: "Reasons for Initializing Flip Flops")

at least one selectable behavior module, which is linkable to a circuit module, and which when executed results in performing the following acts:
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

The reference teaches that "If you want to set an internal node called DRAM_CONTROLX to a value of '1', for example, you can create a small Verilog block in the Verilog test-fixture (.TF) file used to simulate the design. Here is an example ..." The example shows that the behavior module is separate from the module representing the electrical connections in the flip-flop.

forcing an initial forced logic state on the node;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

after forcing, releasing the node from the initial forced logic state if a predetermined condition is met, which enables a simulation program to change a logic state of the node and
(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

monitoring the node after the node has been released; and

providing an indication, in response to the monitoring, when the node is in a preselected condition.

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may never leave the unknown state unless it is manually initialized." There is always a

possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

60. QuickNote #61 teaches the limitations of Claim 20:

20. (Currently Amended) A simulation module of a simulation program, the simulation module comprising:

an input means for inputting an initial node condition into a simulated circuit node of a circuit module linked with the simulation module, wherein the simulated circuit node represents a simulated electrical connection point of the circuit module;
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

The reference teaches that "If you want to set an internal node called DRAM_CONTROLX to a value of '1', for example, you can create a small Verilog block in the Verilog test-fixture (.TF) file used to simulate the design. Here is an example ..." The example shows that the behavior module is separate from the module representing the electrical connections in the flip-flop.

a conveying means for conveying the initial node condition to the simulated circuit node;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

release means for releasing the simulated circuit node from the initial node condition upon satisfaction of a condition, wherein releasing the simulated circuit node <u>occurs after conveying and </u>enables the simulation program to change a logic state of the simulated circuit node;
(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

a monitoring means for monitoring the simulated circuit node for a node condition <u>after releasing the simulated circuit node from the initial node condition</u>; and

an output means, responsive to the monitoring means, for outputting an indication when the node condition is in an undesirable state.

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may

never leave the unknown state unless it is manually initialized." There is always a possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

61. QuickNote #61 teaches the limitations of Claim 24:

24. (Currently Amended) A computerized system for initializing and monitoring a simulated circuit node, the system comprising:

a circuit simulation tool;
(See QuickNote #61, especially: "Reasons for Initializing Flip Flops")

at least one selectable circuit module, which when executed simulates a circuit having the simulated circuit node, wherein the simulated circuit node represents a simulated electrical connection point of the circuit; and
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

at least one selectable behavior module, which is linkable to a circuit module, and which includes
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

The reference teaches that "If you want to set an internal node called DRAM_CONTROLX to a value of '1', for example, you can create a small Verilog block in the Verilog test-fixture (.TF) file used to simulate the design. Here is an example ..." The example shows that the behavior module is separate from the module representing the electrical connections in the flip-flop.

a first input means for inputting an initial node condition;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

a conveying means for conveying the initial node condition to the simulated circuit node;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

a release means for releasing the initial node condition, wherein releasing the initial node condition <u>occurs after conveying and</u> enables the circuit simulation tool to change a logic state of the simulated circuit node;
(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

a monitoring means for monitoring the simulated circuit node for a node condition after releasing the initial node condition;

a first output means for outputting an indication when the node condition is in an undesirable state, in response to the monitoring;

a second input means for inputting a simulation run time; and

a second output means for outputting a final node condition at completion of the simulation run time.

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may never leave the unknown state unless it is manually initialized." There is always a possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

62. QuickNote #61 teaches the limitations of Claim 25:

25. (Previously Presented) An HDL initial condition module comprising:

a means for forcing a logic level on a simulated circuit node;
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

a means for maintaining the logic level of the simulated circuit node until a release condition is met, wherein
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

the simulated circuit node represents a simulated electrical connection point of a simulated circuit and the simulated circuit is produced by an HDL circuit module that is linkable to the HDL initial condition module, and
(See QuickNote #61, especially: 1st para. of "Using the simulator to set initial values for flip-flops")

The reference teaches that "If you want to set an internal node called DRAM_CONTROLX to a value of '1', for example, you can create a small Verilog block in the Verilog test-fixture (.TF) file used to simulate the design. Here is an

example ..." The example shows that the behavior module is separate from the module representing the electrical connections in the flip-flop.

a simulation program is able to change a logic state of the simulated circuit node after the release condition is met.

(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

<u>a means for monitoring the simulated circuit node after the simulated circuit node has been released; and</u>

<u>a means for providing an indication, in response to the monitoring, when the simulated circuit node is in a preselected condition.</u>

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may never leave the unknown state unless it is manually initialized." There is always a possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

63. QuickNote #61 teaches the limitations of Claim 28:

28. (Currently Amended) An HDL initial condition module comprising:

an initial condition release means, which enables a simulation program to change a logic state of a simulated circuit node after a release condition is met, wherein the simulated circuit node represents a simulated electrical connection point of a simulated circuit, and the simulated circuit is produced by an HDL circuit module that is linkable to the HDL initial condition module; and

(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

However, QuickNote #61 does not expressly teach the following limitations:

a simulated circuit node error detection means, which monitors the simulated circuit node for a node condition.

Official Notice is given that it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform these steps, because QuickNote #61 teaches (p.1, 1st para.) that "Another reason for initializing flip-flops is that a free-running toggle flip-flop or divider circuit may never leave the unknown state unless it is manually initialized." There is always a possibility that once a node is released, it "relapses" into an unknown state, in which case some sort of error message would be beneficial.

64. QuickNote #61 teaches the limitations of Claim 29:

29. (Currently Amended) An HDL initial condition module comprising:

means for maintaining a logic level of a simulated circuit node for a predetermined period of time, wherein the simulated circuit node represents a simulated electrical connection point of a simulated circuit, and the simulated circuit is produced by an HDL circuit module that is linkable to the HDL initial condition module; and
(See QuickNote #61, especially: "force" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

means for releasing an initial condition after a release condition is met, wherein releasing the initial condition enables a simulation program to change the logic level of the simulated circuit node, and wherein the predetermined period of time is a simulation run time defined by an HDL simulation executable program.
(See QuickNote #61, especially: "release" command in Examples 1 and 2 of "Using the simulator to set initial values for flip-flops")

65. Claims 9-11 and 34-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over IEEE 1364 in view of IEEE 1164 and further in view of Smith and further in view of Berman.

66. In regards to Claim 9, IEEE 1364 teaches the functionalities of the unknown state (Section 3, p.1), monitoring the state (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), looping statements with conditions (Section 9, pp.111-114), and forcing nodes (Section 9, pp.104-106).

However, IEEE 1364 does not expressly teach the limitations of Claim 9:

9. (Currently Amended) The method of claim 8, wherein the initial node condition is forced again if the testing indicates that the node has an unknown logic value.

On the other hand, IEEE Std. 1164 (pp.4-5) teaches the truth tables of nine different signals (U, X, 0, 1, Z, W, L, H, and '-'), where 'X' represents a "forcing unknown", while a 'W' represents a "weak unknown". The truth tables on pp.4-5 show that these signals remain "unknown" unless they go through an "AND" gate together with an '0' signal, or go through an "OR" gate with a '1' signal.

In other words, the truth tables in IEEE 1164 teach that the signals remain unknown unless they are "forced" to a known value. Therefore, if at at some point during testing or simulation, a signal's logic value becomes unknown, it is inherent that the only way to continue to analyze it is to "force" it again to a known value .

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Smith teaches that: "Hardware structure can be modeled equally effectively in both VHDL and Verilog. ... The choice of which to use is not therefore based solely on technical capability but on: (1) personal preference, (2) EDA tool availability, (3) commercial, business and marketing issues" (See Smith, p.771, "3. VHDL/Verilog compared & contrasted").

Moreover, it would have been obvious of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE

1164, because Berman teaches that: "... most current needs for VHDL/Verilog interoperability are driven by ASIC library availability in Verilog and system level component availability in VHDL." (See Berman, p.142, "2.0 Interoperability Requirements").

67. In regards to Claim 10, IEEE 1364 teaches the functionalities of the unknown and valid logic states (Section 3, p.1), monitoring the state (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), looping statements with conditions (Section 9, pp.111-114), and forcing nodes (Section 9, pp.104-106).

However, IEEE 1364 does not expressly teach the limitations of Claim 10:

10. (Currently Amended) The method of claim 9, wherein the initial node condition is forced and simulation is repeated until the node has a valid logic value.

On the other hand, IEEE Std. 1164 (pp.4-5) teaches the truth tables of nine different signals (U, X, 0, 1, Z, W, L, H, and '-'), where 'X' represents a "forcing unknown", while a 'W' represents a "weak unknown". The truth tables on pp.4-5 show that these signals remain "unknown" unless they go through an "AND" gate together with an '0' signal, or go through an "OR" gate with a '1' signal.

In other words, the truth tables in IEEE 1164 teach that the signals remain unknown unless they are "forced" to a known value. Therefore, if at at some point during testing or simulation, a signal's logic value becomes unknown, it is inherent that the only way to continue to analyze it is to "force" it again to a known value. It is also inherent that if the "forcing" is unsuccessful, the signal will remain unknown, and analysis will be futile, unless the "forcing" is repeated until it is successful.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Smith teaches that: "Hardware structure can be modeled equally effectively in both VHDL and Verilog. ... The choice of which to use is not therefore based solely on technical capability but on: (1) personal preference, (2) EDA tool availability, (3) commercial, business and marketing issues" (See Smith, p.771, "3. VHDL/Verilog compared & contrasted").

Moreover, it would have been obvious of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Berman teaches that: "... most current needs for VHDL/Verilog interoperability are driven by ASIC library availability in Verilog and system level component availability in VHDL." (See Berman, p.142, "2.0 Interoperability Requirements").

68. In regards to Claim 11, IEEE 1364 teaches the functionalities of the unknown and valid logic states (Section 3, p.1), monitoring the state (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), looping statements with conditions (Section 9, pp.111-114), and forcing nodes (Section 9, pp.104-106).

However, IEEE 1364 does not expressly teach the limitations of Claim 11:

11. (Currently Amended) The method of claim 10, wherein monitoring only occurs after the node has a valid logic value.

On the other hand, IEEE Std. 1164 (pp.4-5) teaches the truth tables of nine different signals (U, X, 0, 1, Z, W, L, H, and '-'), where 'X' represents a "forcing unknown", while a 'W' represents a "weak unknown". The truth tables on

pp.4-5 show that these signals remain “unknown” unless they go through an “AND” gate together with an ‘0’ signal, or go through an “OR” gate with a ‘1’ signal.

In other words, the truth tables in IEEE 1164 teach that the signals remain unknown unless they are “forced” to a known value. Therefore, if at some point during testing or simulation, a signal’s logic value becomes unknown, it is inherent that the only way to continue to analyze it is to “force” it again to a known value. It is also inherent that monitoring and analysis will be futile until such time when the “forcing” of the signal successful.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Smith teaches that: “Hardware structure can be modeled equally effectively in both VHDL and Verilog. ... The choice of which to use is not therefore based solely on technical capability but on: (1) personal preference, (2) EDA tool availability, (3) commercial, business and marketing issues” (See Smith, p.771, “3. VHDL/Verilog compared & contrasted”).

Moreover, it would have been obvious of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Berman teaches that: “... most current needs for VHDL/Verilog interoperability are driven by ASIC library availability in Verilog and system level component availability in VHDL.” (See Berman, p.142, “2.0 Interoperability Requirements”).

69. In regards to Claim 34, IEEE 1364 teaches the functionalities of a simulated HDL circuit device (Section 1, pp. iii-iv, 1-4), HDL modules (Section 12, pp.135-139), nodes (Section 6, pp.50-52), unknown and valid logic states (Section 3, p.1), assigning/de-assigning and forcing/releasing node values (Section 9, pp.104-106), monitoring node values (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), and outputting node values (Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;), outputting an undesirable state (Section 9, p.111), condition statements (Section 9, pp.106-107), and looping statements with conditions (Section 9, pp.111-114)

However, IEEE 1364 does not expressly teach that a simulation is repeated ("continuing in phase one") until the node has a valid logic value.

On the other hand, IEEE Std. 1164 (pp.4-5) teaches the truth tables of nine different signals (U, X, 0, 1, Z, W, L, H, and '-'), where 'X' represents a "forcing unknown", while a 'W' represents a "weak unknown". The truth tables on pp.4-5 show that these signals remain "unknown" unless they go through an "AND" gate together with an '0' signal, or go through an "OR" gate with a '1' signal.

In other words, the truth tables in IEEE 1164 teach that the signals remain unknown unless they are "forced" to a known value. Therefore, if at some point during testing or simulation, a signal's logic value becomes unknown, it is inherent that the only way to continue to analyze it is to "force" it again to a

known value. It is also inherent that monitoring and analysis will be futile until such time when the "forcing" of the signal successful.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Smith teaches that: "Hardware structure can be modeled equally effectively in both VHDL and Verilog. ... The choice of which to use is not therefore based solely on technical capability but on: (1) personal preference, (2) EDA tool availability, (3) commercial, business and marketing issues" (See Smith, p.771, "3. VHDL/Verilog compared & contrasted").

Moreover, it would have been obvious of ordinary skill in the art at the time the invention was made to modify the teachings of IEEE 1364 with those of IEEE 1164, because Berman teaches that: "... most current needs for VHDL/Verilog interoperability are driven by ASIC library availability in Verilog and system level component availability in VHDL." (See Berman, p.142, "2.0 Interoperability Requirements").

70. IEEE 1364 teaches the limitations of Claim 35:

35. (Original) The method of claim 34, wherein simulation completion is a user defined time period.

(See IEEE 1364, especially: Sections 9.2, 9.3, pp.103, and 106-107 and pp.114-119; Section 14, pp.179-190, 183-196, 202-204; Section 18, pp.234, 246-247;)

71. Claims 31-32 are rejected under 35 U.S.C. 103(a) as being unpatentable over IEEE 1364 in view of *In re Kuhle*.

72. In regards to Claim 31, IEEE 1364 teaches a simulated HDL circuit device (Section 1, pp. iii-iv, 1-4), HDL modules (Section 12, pp.135-139), nodes (Section 6, pp.50-52), assigning/forcing node values, and condition statements (Section 9, pp.104-106), monitoring node values (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), and outputting node values (Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;), and outputting an undesirable state (Section 9, p.111).

While IEEE 1364 does not expressly teach the use of <u>exactly two</u> HDL modules for assigning, monitoring, or outputting node values, IEEE 1364 teaches (see Section 12) the implementation of an undefined number of modules, and more specifically, it teaches (see Section 11, p.125) that:

> Tasks and functions provide the ability to execute common procedures from several different places in a description. They also provide a means of breaking up large procedures into smaller ones to make it easier to read and debug the source descriptions. This section describes the differences between tasks and functions, describes how to define and invoke tasks and functions, and presents examples of each.

Examiner therefore finds the distribution of the HDL functionalities into exactly two modules to be a matter of design choice. See MPEP §2144.04, specifically *In re Kuhle*, 526 F.2d 553, 188 USPQ 7 (CCPA 1975).

73. In regards to Claim 32, IEEE 1364 teaches a simulated HDL circuit device (Section 1, pp. iii-iv, 1-4), HDL modules (Section 12, pp.135-139), nodes (Section 6, pp.50-52), assigning/de-assigning and forcing/releasing node values, and condition statements (Section 9, pp.104-106), monitoring node values (Section 9.5.2, p.111; Section 14, pp.179-180; Section 18, pp.234, 246-247;), and

outputting node values (Section 9, pp.110-111; Section 14, pp.179-183; Section 18, pp.234, 246-247;), and outputting an undesirable state (Section 9, p.111).

While IEEE 1364 does not expressly teach the use of exactly three HDL modules for assigning, monitoring, or outputting node values (Section 12 enables the implementation of an undefined number of modules), Section 11 teaches (p.125) that:

> Tasks and functions provide the ability to execute common procedures from several different places in a description. They also provide a means of breaking up large procedures into smaller ones to make it easier to read and debug the source descriptions. This section describes the differences between tasks and functions, describes how to define and invoke tasks and functions, and presents examples of each.

Examiner finds the distribution of the HDL functionalities into exactly three modules to be a matter of design choice. See MPEP §2144.04, specifically *In re Kuhle*, 526 F.2d 553, 188 USPQ 7 (CCPA 1975).

Response to Amendment filed 11/24/04

Re: Request for Telephonic Interview

74. The Examiner has noted the Applicants' request for an interview.
75. For the sake of compact prosecution, this Office Action is the Examiner's response to the new amendments and arguments presented by the Applicants in their RCE amendment filed 11/24/04. The Examiner will grant an interview to discuss this Office Action, if contacted by the Applicants' council.

<u>Re: Claim Rejections - 35 USC § 102</u>

76. In regards to Claims 1-8, 12-30, and 33, Applicants argue (Amendment, p.11), that "IEEE 1364 does not describe the arrangements of processes, apparatus, or means elements claimed in the present application", and request that the Examiner provide citations in the IEEE 1364 reference that show the claim elements arranged as set forth in the claims.

Examiner notes that in the previous Office Action, each limitation was referenced to specific citations in the IEEE 1364 reference. However, in light of Applicants' request, Examiner has provided more details in the rejections.

77. Moreover, Examiner finds that the Applicants' repetition of the claim language (Amendment, pp.12-13) fails to comply with 37 CFR 1.111(b), because this amounts to a general allegation that the claims define a patentable invention without specifically pointing out <u>how</u> the language of the claims patentably distinguishes them from the references.

<u>Re: Claim Rejections - 35 USC § 103</u>

78. In the previous Office Action, Claims 9-11 were rejected under IEEE 1364 in view of Official Notice. The previous Office Action, in the arguments section, gave support to the Official Notice included in the rejection of Claims 9-11 by referring the Applicants to the IEEE Std 1164-1993 reference. These rejections have been

rewritten in this Office Action to formally replace the Official Notice with the IEEE 1164 reference.

The Applicants argue (Amendment, p.14) that "This reference includes truth tables ... Applicant disagrees that these truth tables relate to forcing or re-forcing a node condition. They are merely tables indicating inputs and outputs to gates." Examiner respectfully disagrees, and has modified the rejections to address this argument.

The Applicants also argue that (Amendment p.14) that the combination of the teachings of the IEEE Std 1164 and IEEE 1364 is improper because one is for VHDL and the other is for Verilog. Applicants argue that "First, if the languages are competitors, there would be no motivation to combine and, in fact, there would be a motivation away from combining the two languages, because they are incompatible and use different syntaxes. In addition, no suggestion or motivation to combine these references can be found in the references themselves." Examiner respectfully disagrees, and has modified the rejections to address this argument.

The Applicants also assert (Amendment, p.14) that "Still further, IEEE 1364-1995 and IEEE 1164-1993, either singularly or in combination, do not teach or suggest all the features of Claims 9-11, particularly the arrangement of the features in these claims." Examiner finds that this unsubstantiated assertion fails to comply with 37 CFR 1.111(b), because it amounts to a general allegation that

the claims define a patentable invention without specifically pointing out <u>how</u> the language of the claims patentably distinguishes them from the references.

79. In regards to Claims 31 and 32, Examiner has replaced the Official Notice rejections with references to *In re Kuhle*.

Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ayal I. Sharon whose telephone number is (571) 272-3714. The examiner can normally be reached on Monday through Thursday, and the first Friday of a biweek, 8:30 am – 5:30 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin Teska can be reached at (571) 272-3716.

Any response to this office action should be faxed to (703) 872-9306 or mailed to:

Director of Patents and Trademarks
Washington, DC 20231

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Tech Center 2100 Receptionist, whose telephone number is (571) 272-2100.

Ayal I. Sharon

Art Unit 2123

February 9, 2005

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER